METHOD AND SYSTEM FOR ENHANCED DATA SEARCHING

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to a method and system for searching for information in a data set, and, in particular, to methods and systems for syntactically indexing and searching data sets to achieve greater search result accuracy.

Background

Often times it is desirable to search large sets of data, such as collections of millions of documents, only some of which may pertain to the information being sought. In such instances it is difficult to either identify a subset of data to search or to search all data yet return only meaningful results. Several search techniques have been used to support searching large sets of data, none of which have been able to attain a high degree of accuracy of search results due to their inherent limitations.

One common technique is that implemented by traditional keyword search engines. Data is searched and results are generated based on matching one or more words or terms designated as a query. The results are returned because they contain a word or term that matches all or a portion of one or more keywords that were submitted to the search engine as the query. Some keyword search engines additionally support the use of modifiers, operators, or a control language that specifies how the keywords should be combined in a search. For example, a query might specify a date filter to be used to filter the returned results. In many traditional keyword search engines, the results are returned ordered, based on the number of matches found within the data. For example, a keyword search against Internet websites typically returns a list of sites that contain one or more of the submitted keywords, with the sites with the most matches appearing at the top of the list. Accuracy of search results in these systems is thus presumed to be associated with frequency of occurrence.

One drawback to traditional search engines is that they don't return data that doesn't match the submitted keywords, even though it may be relevant. For example, if a user is searching for information on what products a particular country imports, data that refers to the country as a "customer" instead of using the term "import" would be missed if the submitted query specifies "import" as one of the keywords, but doesn't specify the term "customer." (*E.g.*, The sentence "Argentina is a customer of the Acme Company" would be missed.) Ideally, a user would be able to submit a query in the form of a question and receive back a set of results that were accurate based on the meaning of the query – not just on the specific terms used to phrase the question.

Natural language parsing provides technology that attempts to understand and identify the syntactical structure of a language. Natural language parsers have been used to identify the parts of speech of each term in a submitted sentence to support the use of sentences as natural language queries. They have been used also to identify text sentences in a document that follow a particular part of speech pattern; however, these techniques fall short of being able to produce meaningful results when the documents do not follow such patterns. The probability of a sentence falling into a class of predefined sentence templates or the probability of a phrase occurring literally is too low to provide meaningful results. Failure to account for semantic and syntactic variations across a data set, especially heterogeneous data sets, has led to disappointing results.

BRIEF SUMMARY OF THE INVENTION

Embodiments of the present invention provide methods and systems for syntactically indexing and searching data sets to achieve more accurate search results. Example embodiments provide a Syntactic Query Engine ("SQE") that parses, indexes, and stores a data set, as well as processes queries subsequently submitted against the data set. The SQE parses each object in the data set and transforms it into a canonical form that can be searched efficiently using techniques of the present invention. To perform this transformation, the SQE determines the syntactic structure of the data by parsing (or decomposing) each data object into syntactic units, determines the grammatical roles and

relationships of the syntactic units, and represents these relationships in a normalized data structure. A set of heuristics is used to determine which relationships among the syntactic units are important for yielding greater accuracy in the results subsequently returned in response to queries. The normalized data structures are then stored and indexed. The SQE processes queries in a similar fashion by parsing them and transforming them into the same canonical form, which is then used to generate and execute queries against the data set.

In one embodiment, the parsing of each data object into syntactic units is performed by a natural language parser, which generates a hierarchical data structure (*e.g.*, a tree) of syntactic units. In other embodiments, the parser is a module that generates a syntactic structure (or lexical structure) that relates specifically to the objects of the data set. In yet another embodiment, the parser is an existing, off-the-shelf parser, that is modified to perform the SQE transformations of the data set or of queries.

In some embodiments, the canonical form is an enhanced data representation, such as an enhanced sentence representation. In one embodiment, the canonical form comprises a set of tables, which represent grammatical roles of and / or relationships between various syntactic units. In some embodiments, tables are created for the subject, object, preposition, subject/object, noun/noun modifier roles and / or relationships of the syntactic units.

In one embodiment, use of the normalized data structure allows data that appears in multiple and different languages and in different forms to be processed in a similar manner and at the same time. For example, a single query can be submitted against a corpus of documents written in different languages without first translating all of the documents to one language. In another embodiment, the data set may include parts that themselves contain multiple language elements. For example, a single document, like a tutorial on a foreign language, may be written in several languages. In another embodiment, the data set may include objects containing computer language. In yet another embodiment, the data set may include graphical images, with or without surrounding text, bitmaps, film, or other visual data. In yet another embodiment, the data set may include audio data such as music. In summary, the data set may include any data

that can be represented in syntactical units and follows a grammar that associates roles to the syntactical units when they appear in a specified manner, even if the data may not traditionally be thought of in that fashion.

In one embodiment, the processed queries are natural language queries. In other embodiments, the queries are specific representations with form and / or meaning that is specifically related to the objects of the data set.

In one embodiment, the SQE comprises a Query Preprocessor, a Data Set Preprocessor, a Query Builder, a Data Set Indexer, an Enhanced Natural Language Parser ("ENLP"), a data set repository, and, in some embodiments, a user interface. After preprocessing the data set, the SQE parses the data set and determines the syntactic and grammatical roles of each term to generate enhanced data representations for each object (*e.g.*, sentence) in the data set. The SQE indexes and stores these enhanced data representations in the data set repository. Upon subsequently receiving a query, such as a natural language query, the SQE parses the query similarly and searches the stored indexed data set to locate data that contains similar terms used in similar grammatical roles.

In some embodiments, the SQE provides search operators based upon the grammatical roles and relationships of syntactic units of objects of data. For example, some embodiments provide a search that allows designation of the grammatical role of a unit of the query. For example, a term may be designated as a subject or an object of a sentence before it is used to search the data set for similar terms used in similar grammatical roles. In one embodiment, the SQE returns a list of related units (terms) based upon a grammatical role. For example, in response to a query that designates a term as a "subject" of a textual phrase, the SQE returns a list of verbs that appear in phrases that contain the term used as the subject of those phrases. Other embodiments return different parts of speech or terms that appear in particular grammatical roles.

In yet other embodiments, the SQE provides an ability to search for similar sentences in a data set of documents or similar objects, where similar means that the matching sentence contains similar words used in similar grammatical roles or syntactic relationships. In some embodiments, this ability is invoked by selection of a sentence in

data that is returned as a result of another query. In yet other embodiments, the SQE provides an ability to search for similar paragraphs and similar documents. In other embodiments, the SQE provides an ability to search for similar objects.

In some embodiments, the SQE returns results to a query in an order that indicates responsiveness to the query. In some embodiments, the order is based upon the polysemy of terms in the query. In other embodiments, the order is based upon the inverse document frequency of terms in the query. In yet other embodiments, the ordering of results is based upon weightings of the matched results from the data set. In some of these embodiments, the weightings are based upon the degree of matching of a particular part of speech. For example, the weighting may be based upon whether matching sentences contain identical verbs, entailed verbs, or verbs that are defined as close by some other metric, such as frequency or distribution in the data set.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a natural language query and the results returned by an example embodiment of a Syntactic Query Engine.

Figure 2 is an example block diagram of a Syntactic Query Engine.

Figure 3 is an example flow diagram of the steps performed by a Syntactic Query Engine to process data sets and natural language queries.

Figure 4 is an example screen display illustrating general search functionality of an example Syntactic Query Engine user interface.

Figure 5A is an example screen display illustrating a portion of a data set from which a natural language query result was extracted.

Figure 5B is an example screen display illustrating a search similar sentence operation.

Figure 5C is an example screen display illustrating results that correspond to the search similar sentence operation initiated in Figure 5B.

Figure 6 is an example screen display illustrating Syntactic Query Engine results from a natural language query that requests a map.

Figure 7 is an example screen display illustrating a map that corresponds to the query result selected in Figure 6.

Figure 8 is an example screen display illustrating Syntactic Query Engine results from a natural language query that requests a chart.

Figure 9 is an example screen display illustrating a chart that corresponds to the query result selected in Figure 8.

Figure 10 is an example screen display of an advanced search using a natural language query that contains a subject.

Figure 11 is an example screen display illustrating advanced search results from a query that contains a subject.

Figure 12 is an example screen display illustrating a portion of resulting sentences returned by a Syntactic Query Engine when a particular verb is selected from the verb list.

Figure 13 is an example screen display of advanced search functionality using a natural language query that contains a subject and an object.

Figure 14 is an example screen display illustrating advanced search results from a query that contains a subject and an object.

Figure 15 is an example screen display illustrating the designation of programmable attributes in a Syntactic Query Engine.

Figure 16 is a block diagram of the components of an example embodiment of a Syntactic Query Engine.

Figure 17 is a block diagram of the components of an Enhanced Natural Language Parser of an example embodiment of a Syntactic Query Engine.

Figure 18 is a block diagram of the processing performed by an example Enhanced Natural Language Parser.

Figure 19 is a block diagram illustrating a graphical representation of an example syntactic structure generated by the natural language parser component of an Enhanced Natural Language Parser.

Figure 20 is a table illustrating an example enhanced data representation generated by the postprocessor component of an Enhanced Natural Language Parser.

Figure 21 is an example block diagram of data set processing performed by a Syntactic Query Engine.

Figure 22 is an example block diagram of natural language query processing performed by a Syntactic Query Engine.

Figure 23 is an example block diagram of a general purpose computer system for practicing embodiments of a Syntactic Query Engine.

Figure 24 is an example flow diagram of the steps performed by a build_file routine within the Data Set Preprocessor component of a Syntactic Query Engine.

Figure 25 illustrates an example format of a tagged file built by the build_file routine of the Data Set Preprocessor component of a Syntactic Query Engine.

Figure 26 is an example flow diagram of the steps performed by the dissect_file routine of the Data Set Preprocessor component of a Syntactic Query Engine.

Figure 27 is an example flow diagram of the steps performed by a Syntactic Query Engine to process a natural language query.

Figure 28 is an example flow diagram of the steps performed by a preprocess_natural_language_query routine of the Query Preprocessor component of a Syntactic Query Engine.

Figure 29 is an example block diagram showing the structure of an example Data Set Repository of a Syntactic Query Engine.

Figure 30 is an example flow diagram of the steps performed by a parse_sentence routine of the Enhanced Natural Language Parser component of a Syntactic Query Engine.

Figure 31 is an example flow diagram of the steps performed by a determine_grammatical_roles subroutine within the parse_sentence routine of the Enhanced Natural Language Parser.

Figure 32 is an example flow diagram of the steps performed by a generate_subject_structure subroutine of the determine_grammatical_roles routine.

Figure 33 is an example flow diagram of the steps performed by a generate_object_structure subroutine of the determine_grammatical_roles routine.

Figure 34 is an example flow diagram of the steps performed by a generate_subject_modifier subroutine of the determine_grammatical_roles routine.

Figure 35 is an example flow diagram of the steps performed by a generate_generalized_subject_object subroutine of the determine_grammatical_roles routine.

Figure 36A is a graphical representation of an example parse tree generated by a natural language parser component of an Enhanced Natural Language Parser.

Figure 36B is an illustration of an enhanced data representation of an example natural language query generated by an Enhanced Natural Language Parser.

Figure 37 is an example flow diagram of the steps performed by a construct_output_string routine of the Enhanced Natural Language Parser.

Figure 38 is an example flow diagram of the steps performed by an index_data routine of the Data Indexer component of a Syntactic Query Engine.

Figures 39A and 39B are example flow diagrams of the steps performed by a build_query routine within the Query Builder component of a Syntactic Query Engine.

DETAILED DESCRIPTION OF THE INVENTION

Embodiments of the present invention provide methods and systems for syntactically indexing and searching data sets to achieve more accurate search results. Example embodiments provide a Syntactic Query Engine ("SQE") that parses, indexes, and stores a data set, as well as processes queries subsequently submitted against the data set. The SQE parses each object in the data set and transforms it into a canonical form that can be searched efficiently using techniques of the present invention. To perform this transformation, the SQE determines the syntactic structure of the data by parsing (or decomposing) each data object into syntactic units, determines the grammatical roles and relationships of the syntactic units, and represents these relationships in a normalized data structure. A set of heuristics is used to determine which relationships among the syntactic units are important for yielding greater accuracy in the results subsequently returned in response to queries. The normalized data structures are then stored and indexed. The SQE processes queries in a similar fashion by parsing them and transforming them into the same canonical form, which is then used to generate and execute queries against the data set..

In one embodiment, the SQE includes, among other components, a data set repository and an Enhanced Natural Language Parser ("ENLP"). The ENLP parses the initial data set and the natural language queries and determines the syntactic and grammatical roles of terms in the data set / query. Then, instead of matching terms found in the query with identical terms found in the data set (as done in typical keyword search engines), the SQE locates terms in the data set that have similar grammatical roles to the grammatical roles of similar terms in the original query. In this manner, the SQE is able to achieve more contextually accurate search results more frequently than using traditional search engines.

One skilled in the art will recognize that, although the techniques are described primarily with reference to text-based languages and collections of documents, the same techniques may be applied to any collection of terms, phrases, units, images, or other objects that can be represented in syntactical units and that follow a grammar that defines and assigns roles to the syntactical units, even if the data object may not

traditionally be thought of in that fashion. Examples include written or spoken languages, for example, English or French, computer programming languages, graphical images, bitmaps, music, video data, and audio data. Sentences that comprise multiple words are only one example of a phrase or collection of terms that can be analyzed, indexed, and searched using the techniques described herein.

In addition, the term "natural language query" is used to differentiate the initial query from subsequent data queries created by an SQE in the process of transforming the initial query into a set of data-set-specific queries (*e.g.*, database queries) that are executed against the indexed data set. One skilled in the art will recognize, however, that the form and content of the initial query will relate to and depend upon the form of objects in the data set—*i.e.*, the language of the data set.

The Syntactic Query Engine is useful in a multitude of scenarios that require indexing, storage, and/or searching of, especially large, data sets, because it yields results to data set queries that are more contextually accurate than other search engines. In a text-based, document environment, the SQE identifies the syntax of a submitted natural language query or sentence within a data set (*e.g.*, which terms are nouns, adjectives, verbs, and other parts of speech, and the relationships between the terms), determines which terms are less likely to produce meaningful results (*e.g.*, words like "the", "a", and "who"), and ignores these terms. For example, given the natural language query,

> What types of scientific research does the Department of Defense fund?

the SQE identifies "scientific" as an adjective, "research" as a noun, "Department of Defense" as a noun phrase, and "fund" as a verb. The other terms in the sentence are ignored as being less meaningful. Based on the identified syntax, the SQE determines the grammatical role of each meaningful term (*e.g.*, whether the term is a subject, object, or governing verb). Based upon a set of heuristics, the SQE uses the determined roles to generate one or more data queries from the natural language query to execute against the data set, which has been indexed and stored previously using a similar set of heuristics. For example, in the query above, "Department of Defense" is determined to be a subject of

the sentence, "fund" is determined to be the governing verb of the sentence, and "scientific" and "research" are both determined to be objects of the sentence. Based on the determined grammatical roles, the SQE is able to generate an enhanced data representation and, hence, data queries that tend to return more contextually accurate results because the data set has been transformed to a similar enhanced data representation. Specifically, rather than identifying data in the data set that matches terms or parts of terms in the query (like traditional keyword search engines), the SQE executes the data queries to return data that contains similar terms used in similar grammatical roles to those terms and grammatical roles in the submitted natural language query. In summary, the SQE uses its determination of the grammatical roles of terms to transform a syntactic representation of data in the data set and in queries submitted against the data to a canonical representation that can be efficiently compared.

Figure 1 shows a natural language query and the results returned by an example embodiment of a Syntactic Query Engine. The example natural language query,

Does Argentina import or export gas?

shown in query box 102, when executed against a previously indexed data set, returns results in results area 103 that relate to Argentina's importation and exportation of gas, even though the terms "import" and "export" do not appear in all of the records of the results. Thus, the SQE is able to make what one would consider greater contextual associations between the designated natural language query and the data set than a traditional keyword search would provide. This capability is due to the SQE's ability to index data sets and perform syntactical searches based on determined grammatical roles of and relationships between terms in the query and terms within sentences in the data set, as opposed to keyword searches, yielding a more effective search tool. The SQE is thus able to "recognize" that certain terms in a data set may be relevant in relation to a submitted query simply because of their grammatical roles in the sentence. For example, the first sentence returned 104 refers to Argentina as a "customer" as opposed to an "importer." This sentence may not have been returned as a result of the shown natural language query

had a traditional keyword search been performed instead, because "customer" is not identical or a part of the term "importer."

Figure 2 is an example block diagram of a Syntactic Query Engine. A document administrator 202 adds and removes data sets (for example, sets of documents), which are indexed and stored within a data set repository 204 of the SQE 201. A subscriber 203 to a document service submits natural language queries to the SQE 201, typically using a visual interface. The queries are then processed by the SQE 201 against the data sets stored in the data set repository 204. The query results are then returned to the subscriber 203. In this example, the SQE 201 is shown implemented as part of a subscription document service, although one skilled in the art will recognize that the SQE may be made available in many other forms, including as a separate application/tool, integrated into other software or hardware, for example, cell phones, personal digital assistants ("PDA"), or handheld computers, or associated with other types of services. Additionally, although the example embodiment is shown and described as processing data sets and natural language queries that are in the English language, as discussed earlier, one skilled in the art will recognize that the SQE can be implemented to process data sets and queries of any language, or any combination of languages.

Figure 3 is an example flow diagram of the steps performed by a Syntactic Query Engine to process data sets and natural language queries. In step 301, the SQE receives a data set, for example, a set of documents. In step 302, the SQE preprocesses the data set to ensure a consistent data format. In step 303, the SQE parses the data set, identifying the syntax and grammatical roles of terms within the data set and transforming the data to a normalized data structure. In step 304, the SQE stores the parsed and transformed data set in a data set repository. After a data set is stored, the SQE can process natural language queries against the data set. In step 305, the SQE receives a natural language query, for example, through a user interface. In step 306, the SQE preprocesses the received natural language query, formatting the query as appropriate to be parsed. In step 307, the SQE parses the formatted query, identifying the syntactic and grammatical roles of terms in the query and transforming the query into a normalized data structure.

Using the parsed query, in step 308, the SQE generates and submits data queries (*e.g.*, SQL statements) against the data set stored in the data set repository. Finally, in step 309, the SQE returns the results of the natural language query, for example, by displaying them through a user interface.

Figures 4 through 15 are example screen displays of an example Syntactic Query Engine user interface for submitting natural language queries and viewing query results. Figure 4 is an example screen display illustrating general search functionality of an example Syntactic Query Engine user interface. The general search functionality allows a user, for example, to submit a natural language query in the form of a sentence, which may be a statement or a question. The user enters a query in query box 401 and, using the search button 403, submits the query to the SQE, for example, the SQE of Figure 2. The SQE displays the query results in results area 405. Each result that is returned by the SQE after performing a general search is a sentence extracted from the data set. Depending on the data set, other information may also be displayed with each result. For example, additional information displayed with the results shown in Figure 4 include the document title (*e.g.*, "Foreign reserves & the exchange rate"), the name of the country the document relates to (*e.g.*, "Somalia"), the date associated with the document (*e.g.*, "[28-DEC-1997]"), and the location (*e.g.*, directory) where the original source document is stored. One skilled in the art will recognize that depending on the type of data that comprises the data set, different types of related information may be displayed as part of a result in a result set. Each displayed result is also a link that, when selected, causes the SQE user interface to display a larger portion of the data set from which the selected result was extracted. For example, if the sentence 406 shown as,

> The now defunct Central Bank of Somalia suffered a setback after the fall of Mr. Siad Barre in 1991 when a reported $70m in foreign exchange disappeared,

is selected, the SQE will display a portion of the source document 407, labeled,

> Foreign reserves & the exchange rate,

from which that sentence was retrieved. Although this example refers to a "user" submitting natural language queries, one skilled in the art will recognize that natural language queries may be submitted to an SQE through means other than user input. For example, an SQE may receive input from another program executing on the same computer or, for example, across a network or from a wireless PDA device.

Figure 5A is an example screen display illustrating a portion of a data set from which a natural language query result was extracted. The portion displayed in text area 5A05 typically reflects the selected sentence that resulted from the query as well as a number of surrounding sentences in the document. Additional options are also available while viewing portions of the data set. For example, the user may select a sentence in text area 5A05 and select option "Sentences" 5A01 from the Search Similar Menu 5A04, causing the SQE to perform a syntactic search (using the search and indexing techniques described herein) against the data set using the selected sentence as a new natural language query. The search similar sentence functionality is described in detail with reference to Figures 5B and 5C. Alternatively, a user may select an entire paragraph and select option "Paragraphs" 5A02 from the Search Similar Menu 5A04. Selecting the Search Similar Paragraphs option causes the SQE to perform a search against the data set to return other paragraphs within the data set that contain content similar to the selected paragraph. The user may also select option "Documents" 5A03 from the Search Similar Menu 5A04, causing the SQE to perform a search against the data set to return other documents within the data set that contain content similar to the selected document. In an exemplary embodiment of an SQE, the paragraph and document similarity searches are preferably performed using latent semantic regression techniques as described in U.S. Patent Application No. _____, filed on September 25, 2001 and entitled "An Inverse Inference Engine for High Performance Web Search," which is a continuation-in-part of U.S. Application No. 09/532,605 filed March 22, 2000, and claims priority from U.S. Provisional Application No. 60/235,255, filed on September 25, 2000, all of which are incorporated herein by reference in their entirety. One skilled in the art will recognize that other types of searches, including syntactic searches as described herein, may be

implemented for the "Sentences," "Documents," and "Paragraphs" options. For example, a keyword search or one or more syntactic searches may be used.

Figure 5B is an example screen display illustrating a search similar sentence operation. In Figure 5B, a sentence is selected within a portion of a data set from which a natural language query result was extracted. Selecting the sentence 5B06 and then selecting option "Sentences" 5B01 from the Search Similar Menu 5B04 causes the SQE to perform a syntactic search against the data set, returning results that are similar to the selected sentence 5B06. Figure 5C is an example screen display illustrating results that correspond to the search similar sentence operation initiated in Figure 5B. From the results displayed, one can see that the resulting sentences relate in a manner that goes beyond word or pattern matching.

Although discussed herein primarily with respect to documents, the SQE supports data sets comprising a variety of objects and data formats and is not limited to textual documents. For example, a data set may comprise text documents of various formats, with or without embedded non-textual entities (*e.g.*, images, charts, graphs, maps, etc.), as well as documents that are in and of themselves non-textual entities. Figures 6 through 9 illustrate support of natural language queries that request specific non-textual data. One skilled in the art will recognize that support for other format combinations of stored and requested data are contemplated.

Figure 6 is an example screen display illustrating Syntactic Query Engine results from a natural language query that requests a map. When the user selects one of the returned results, for example, the "Angola [icon]" 608, the requested map is displayed. Figure 7 is an example screen display illustrating a map that corresponds to the query result selected in Figure 6.

Figure 8 is an example screen display illustrating Syntactic Query Engine results from a natural language query that requests a chart. When the user selects one of the returned results, for example, the "China [icon] " 808, the requested chart is displayed. Figure 9 is an example screen display illustrating a chart that corresponds to the query result selected in Figure 8.

In addition to queries that are formulated as one or more sentences, the SQE supports searching based upon designated syntactic and/or grammatical roles. The SQE advanced search functionality supports natural language queries that specify one or more terms, each associated with a grammatical role or part of speech.

Figure 10 is an example screen display of an advanced search using a natural language query that contains a subject. From this screen display, a user may enter a word or phrase as a subject 1001 and/or as an object 1002, the "subject" and "object" each being a grammatical role. Selecting the Search button 1003 submits the query to the SQE. In the example shown, a user enters "Bill Clinton" as the subject. When the user selects the Search button 1003, the SQE performs a syntactic search, returning a list of verbs from sentences within the data set which contain "Bill Clinton" as a subject.

Figure 11 is an example screen display illustrating advanced search results from a query that contains a subject. The results are displayed as a list of the verbs found in sentences within the stored data set in which the designated subject occurs as an identified subject of the sentence. The number in brackets after each listed verb indicates the number of times the designated subject and listed verb are found together in a sentence within the stored data set. For example, the top entry of the middle column 1101 of the verb list indicates that the SQE identified 16 sentences within the data set which contain "Bill Clinton" as a subject and "visit" (or some form of the verb "visit") as the verb. The SQE determines the order in which the verb list is displayed. As illustrated in Figure 11, the verb list may be displayed according to decreasing frequency of occurrence of the designated subject and/or object in the data set. In an alternate embodiment, the verbs may be displayed in alphabetical order. In another embodiment, the verbs may be grouped and/or ordered based on the similarity of meanings among the displayed verbs. The similarity of meanings among multiple verbs can be determined using an electronic dictionary, for example, WordNet. The WordNet dictionary is described in detail in Christiane Fellbaum (Editor) and George Miller (Preface), *WordNet (Language, Speech, and Communication)*, MIT Press, May 15, 1998, which is herein incorporated by reference in its entirety.

When the user selects a verb from the returned list, the SQE returns a set of resulting sentences with the designated subject and selected verb. Figure 12 is an example screen display illustrating a portion of resulting sentences returned by a Syntactic Query Engine when a particular verb is selected from the verb list. In the example shown, the verb "visit" has been selected from the verb list shown in Figure 11, causing the SQE to display the 16 sentences within the data set where "Bill Clinton" is found as a subject and "visit" is found as the verb. The SQE identifies verbs without regard to specific tense, which enhances the contextual accuracy of the search. For example, if the user selects "visit" as the verb, the SQE returns sentences that contain the verb "visited" (*e.g.*, results 1-4 and 6), and sentences that contain the verb "may visit" (*e.g.*, result 5). The results displayed in the results area 1205 are displayed in the same format as the results displayed in the results area 405 of Figure 4. Accordingly, selecting one of the displayed results causes the SQE to display a larger portion of the data set from which the selected result was extracted, as described with reference to Figure 5A.

As described, with reference to Figure 10, a user may designate a subject and/or an object when using the advanced search functionality. Figure 13 is an example screen display of advanced search functionality using a natural language query that contains a subject and an object. In Figure 13, the user designates "US" as a subject in subject field 1301 and "Mexico" as an object in object field 1302. When the user selects the Search button 1303, the SQE performs a syntactic search against the data set for sentences in which "US" appears as an identified subject and "Mexico" appears as an identified object.

Figure 14 is an example screen display illustrating advanced search results from a query that contains a subject and an object. The top half 1406 of results area 1405 lists verbs returned where the designated subject appears as a subject of the sentence and the designated object appears as an object of the sentence. The lower half 1407 of results area 1405 lists verbs returned where the designated object appears as a *subject* of the sentence and the designated subject appears as an *object* of the sentence. Thus, the lower half 1407 displays results of a query using the inverse relationship between the subject and

object. In this specific example, the top half 1406 of results area 1405 lists verbs found in sentences which contain "US" as a subject and "Mexico" as an object. The bottom half 1406 of results area 1408 lists verbs found in sentences which contain "Mexico" as a subject and "US" as an object. Returning results related to the inverse relationship can be useful because sentences within the data set that are contextually accurate results to the natural language query may be overlooked if the inverse relationship is not examined. As similarly described with reference to Figures 11 and 12, when the user selects a verb from the returned list, the SQE returns a set of resulting sentences with the designated subject, designated object, and selected verb. The resulting sentences are displayed in the same format as the results displayed in the results area 405 of Figure 4 and the results displayed in the results area 1205 of Figure 12. Accordingly, selecting one of the displayed sentences causes the SQE to display a larger portion of the data set from which the selected sentence was extracted, as described with reference to Figure 5A.

Although Figures 10-14 illustrate advanced search functionality with reference to natural language queries that specify a subject and/or an object, one skilled in the art will recognize that a multitude of syntactic and grammatical roles, and combinations of syntactic and grammatical roles may be supported. For example, some of the combinations contemplated for support by the advanced search functionality of the SQE are:

subject/object;
subject/verb/object;
subject/verb;
verb/object;
preposition/verb modifier/object;
verb/verb modifier/object;
verb/preposition/object;
verb/preposition/verb modifier/object;
subject/preposition/verb modifier;
subject/preposition/verb modifier/object;
subject/verb/verb modifier/object;
subject/verb/preposition;
subject/verb/preposition/object;
subject/verb/preposition/verb modifier;
subject/ verb/preposition/verb modifier/object; and
noun/noun modifier.

Such support includes locating sentences in which the designated terms appear in the associated designated syntactic or grammatical role, as well as locating, when contextually appropriate, sentences in which the designated terms appear but where the designated roles are interchanged. For example, as described above, it is contextually appropriate to interchange the grammatical roles of a designated subject and a designated object.

In addition to indexing and searching based on grammatical roles, the Syntactic Query Engine may be implemented to recognize any number of programmable attributes in natural language queries and data sets (described in detail as "preferences" with reference to Figure 15). In one embodiment, these attributes are used to filter the results of a syntactic search. Example attributes include the names of countries, states, or regions, dates, and document sections. One skilled in the art will recognize that an unlimited number of attributes may be defined and may vary across multiple data sets. For example, one data set may consist of text from a set of encyclopedias. For such a data set, a "volume" attribute may be defined, where there is one volume for each letter of the alphabet. A second data set may be a single book with multiple chapters. A "chapter" attribute may be defined, for the data set, allowing a user to search specific chapters.

Figure 15 is an example screen display illustrating the designation of programmable attributes in a Syntactic Query Engine. A user designates various attributes on the preferences window 1501. The SQE stores these attributes (preferences) when the user selects the Set Preferences button 1502. The attribute values are used by the SQE as filters when performing subsequent queries. For example, the user may select a specific country as country attribute 1503. When the SQE performs a subsequent natural language query, the results returned are only those sentences found in documents within the data set that relate to the country value specified as country attribute 1503.

A more detailed description of an example SQE illustrating additional user interface screens and example natural language queries and results is included in Appendix A, which is herein incorporated by reference in its entirety.

An SQE as described may perform multiple functions (*e.g.*, data set parsing, data set storage, natural language query parsing, and data query processing) and typically comprises a plurality of components. Figure 16 is a block diagram of the components of an example embodiment of a Syntactic Query Engine. A Syntactic Query Engine comprises a Query Preprocessor, a Data Set Preprocessor, a Query Builder, a Data Set Indexer, an Enhanced Natural Language Parser ("ENLP"), a data set repository, and, in some embodiments, a user interface. The Data Set Preprocessor 1603 converts received data sets to a format that the Enhanced Natural Language Parser 1604 recognizes. The Query Preprocessor 1610 converts received natural language queries to a format that the Enhanced Natural Language Parser 1604 recognizes. The Enhanced Natural Language Parser ("ENLP") 1604, parses sentences, identifying the syntax and grammatical role of each meaningful term in the sentence and the ways in which the terms are related to one another and transforming the sentences into a canonical form—an enhanced data representation. The Data Set Indexer 1607 indexes the parsed data set and stores it in the data set repository 1608. The Query Builder 1611 generates and executes formatted queries (*e.g.*, SQL statements) against the data set indexed and stored in the data set repository 1608.

In operation, the SQE 1601 receives as input a data set 1602 to be indexed and stored. The Data Set Preprocessor 1603 prepares the data set for parsing by assigning a Document ID to each document that is part of the received data set, performing OCR processing on any non-textual entities that are part of the received data set, and formatting each sentence according to the ENLP format requirements. The Enhanced Natural Language Parser ("ENLP") 1604 parses the data set, identifying for each sentence, a set of terms, each term's part of speech and associated grammatical role and transforming this data into an enhanced data representation. The Data Set Indexer 1607 formats the output from the ENLP and sends it to the data set repository 1608 to be indexed and stored. After a data set is indexed, a natural language query 1609 may be submitted to the SQE 1601 for processing. The Query Preprocessor 1610 prepares the natural language query for parsing. The preprocessing may include, for example, spell checking, verifying that there is only

one space between each word in the query, and identifying the individual sentences if the query is made up of more than one sentence. One skilled in the art will recognize that the steps performed by the Data Set Preprocessor or the Query Preprocessor may be modified based on the requirements of the natural language parser. Any preprocessing steps necessary to prepare a data set or a natural language query to be parsed are contemplated for use with techniques of the present invention. The ENLP 1604 then parses the preprocessed natural language query transforming the query into the canonical form and sends its output to the Query Builder. The Query Builder 1611 uses the ENLP 1604 output to generate one or more data queries. Data queries differ from natural language queries in that they are in a format specified by the data set repository, for example, SQL. The Query Builder 1611 may generate one or more data queries associated with a single natural language query. The Query Builder 1611 executes the generated data queries against the data set repository 1608 using well-known database query techniques and returns the data query results as Natural Language Query Results 1612. Note that when the SQE is used within a system that interfaces with a user, the SQE also typically contains a user interface component 1613. The user interface component 1613 interfaces to a user in a manner similar to that shown in the display screens of Figures 4-15.

Figure 17 is a block diagram of the components of an Enhanced Natural Language Parser of an example embodiment of a Syntactic Query Engine. The Enhanced Natural Language Parser ("ENLP") 1701 comprises a natural language parser 1702 and a postprocessor 1703. The natural language parser 1702 identifies, for each sentence it receives as input, the part of speech for each term in the sentence and syntactic relationships between the terms within the sentence. An SQE may be implemented by integrating a proprietary natural language parser into the ENLP, or by integrating an existing off-the-shelf natural language parser, for example, Minipar, available from Nalante, Inc., 245 Falconer End, Edmonton, Alberta, T6R 2V6. The postprocessor 1703 examines the natural language parser 1702 output and, from the identified parts of speech and syntactic relationships, determines the grammatical role played by each term in the sentence and the grammatical relationships between those terms. The postprocessor 1703

then generates an enhanced data representation from the determined grammatical roles and syntactic and grammatical relationships.

Figure 18 is a block diagram of the processing performed by an example Enhanced Natural Language Parser. The natural language parser 1801 receives a sentence 1803 as input, and generates a syntactic structure, such as parse tree 1804. The generated parse tree identifies the part of speech for each term in the sentence and describes the relative positions of the terms within the sentence. The postprocessor 1802 receives the generated parse tree 1804 as input and determines the grammatical role of each term in the sentence and relationships between terms in the sentence, generating an enhanced data representation, such as enhanced sentence representation 1805.

Figure 19 is a block diagram illustrating a graphical representation of an example syntactic structure generated by the natural language parser component of an Enhanced Natural Language Parser. The parse tree shown is one example of a representation that may be generated by a natural language parser. The techniques of the methods and systems of the present invention, implemented in this example in the postprocessor component of the ENLP, enhance the representation generated by the natural language processor by determining the grammatical role of each meaningful term, associating these terms with their determined roles and determining relationships between terms. In Figure 19, the top node 1901 represents the entire sentence, "YPF of Argentina exports natural gas" Nodes 1902 and 1903 identify the noun phrase of the sentence, "YPF of Argentina," and the verb phrase of the sentence, "exports natural gas," respectively. The branches of nodes or leaves in the parse tree represent the parts of the sentence further divided until, at the leaf level, each term is singled out and associated with a part of speech. A configurable list of words are ignored by the parser as "stopwords." The stopword list comprises words that are deemed not indicative of the information being sought. Example stopwords are "a," "the," "and," "or," and "but." In one embodiment, question words (*e.g.*, "who," "what," "where," "when," "why," "how," and "does") are also ignored by the parser. In this example, nodes 1904 and 1905 identify the noun phrase 1902 as a noun, "YPF" and a prepositional phrase, "of Argentina." Nodes 1908 and 1909 divide the

prepositional phrase 1905 into a preposition, "of," and a noun, "Argentina." Nodes 1906 and 1907 divide the verb phrase 1903 into a verb, "exports;" and a noun phrase, "natural gas." Nodes 1910 and 1911 divide the noun phrase 1907 into an adjective, "natural," and a noun, "gas."

Figure 20 is a table illustrating an example enhanced data representation generated by the postprocessor component of an Enhanced Natural Language Parser. This example enhanced data representation comprises nine different ways of relating terms within the sentence that was illustrated in the parse tree of Figure 19. The ways chosen and the number of ways used is based upon a set of heuristics, which may change as more knowledge is acquired regarding syntactic searching. In addition, one skilled in the art will recognize that the selected roles and relationships to be stored may be programmatically determined. In the example shown, row 2001 represents the relationship between "Argentina" as the subject of the sentence and "YPF" as a modifier of that subject. The SQE determines this relationship based on the location of the preposition, "of" between the two related terms in the sentence. Rows 2002 and 2003 represent the relationship between "YPF" as the subject of the sentence and "natural gas" and "gas," respectively, as objects of the sentence. Similarly, rows 2004 and 2005 represent the relationship between "Argentina" as the subject of the sentence and "natural gas" and "gas," respectively, as objects of the sentence. Rows 2006 and 2007, respectively, represent the relationship between the two nouns, "YPF" and "Argentina," each used as a subject and the verb "export." Rows 2008 and 2009 represent the relationship between the verb, "export" and the noun phrase, "natural gas" and the noun, "gas," each used as an object, respectively.

The enhanced data representation is indexed and stored to support the syntactic search functionality of the SQE. The original sentence "YPF of Argentina exports natural gas," will be returned by the SQE as a query result in response to any submitted query that can be similarly represented. For example, "What countries export gas?," "Does Argentina import gas?," and "Is Argentina an importer or exporter of gas?" will all cause the SQE to return to the represented sentence as a result.

The Syntactic Query Engine performs two functions to accomplish effective syntactic query processing. The first is the parsing, indexing, and storage of a data set. The second is the parsing and subsequent execution of natural language queries. These two functions are outlined below with reference to Figures 21 and 22.

Figure 21 is an example block diagram of data set processing performed by a Syntactic Query Engine. As an example, documents that make up a data set 2101 are submitted to the Data Set Preprocessor 2102 (*e.g.*, component 1603 in Figure 16). If the data set comprises multiple files, as shown in Figure 21, the Data Set Preprocessor 2102 creates one tagged file containing the document set. The Data Set Preprocessor 2102 then dissects that file into individual sentences and sends each sentence to the ENLP 2104 (*e.g.*, component 1604 in Figure 16). After the ENLP 2104 parses each received sentence, it sends the generated enhanced data representation of each sentence to the Data Set Indexer 2105 (*e.g.*, component 1607 in Figure 16). The Data Set Indexer 2105 processes and formats the ENLP output, distributing the data to formatted text files. The text files are typically bulk loaded into the data set repository 2107 (*e.g.*, component 1608 in Figure 16). One skilled in the art will recognize that other methods of data set preprocessing, indexing, and storing may be implemented in place of the methods described herein, and that such modifications are contemplated by the methods and systems of the present invention. For example, the Data Set Indexer may insert data directly into the data set repository instead of generating text files to be bulk loaded.

After indexing and storing a data set, the SQE may perform its second function, processing natural language queries against the stored data set. Figure 22 is an example block diagram of natural language query processing performed by a Syntactic Query Engine. As an example, a natural language query 2201 is submitted to the Query Preprocessor 2202 of the SQE. The Query Preprocessor 2202 (*e.g.*, component 1610 in Figure 16) prepares the natural language query for parsing. The preprocessing step may comprise several functions, examples of which may be spell checking, text case verification and/or alteration, and excessive white-space reduction. The specific preprocessing steps performed by the Query Preprocessor 2202 are typically based on the

format requirements of the natural language parser component of the ENLP. The SQE sends the preprocessed query to the ENLP 2204 (*e.g.*, component 1604 in Figure 16). The ENLP parses the query, generating an enhanced data representation of the query 2205, which is sent to the Query Builder 2206 (*e.g.*, component 1611 in Figure 16). This enhanced data representation identifies grammatical roles of and relationships between terms in the query. Using the enhanced data representation 2205, the Query Builder 2206 generates one or more data queries 2207 and executes them against the data set repository 2208. The data query results 2209 are returned to the Query Builder to be returned to the user as natural language query results 2210.

Figure 23 is an example block diagram of a general purpose computer system for practicing embodiments of a Syntactic Query Engine. The computer system 2301 contains a central processing unit (CPU) 2302, Input/Output devices 2303, a display device 2304, and a computer memory (memory) 2305. The Syntactic Query Engine 2320 including the Query Preprocessor 2306, Query Builder 2307, Data Set Preprocessor 2308, Data Set Indexer 2311, Enhanced Natural Language Parser 2312, and data set repository 2315, preferably resides in memory 2305, with the operating system 2309 and other programs 2310 and executes on CPU 2302. One skilled in the art will recognize that the SQE may be implemented using various configurations. For example, the data set repository may be implemented as one or more data repositories stored on one or more local or remote data storage devices. Furthermore, the various components comprising the SQE may be distributed across one or more computer systems including handheld devices, for example, cell phones or PDAs. Additionally, the components of the SQE may be combined differently in one or more different modules. The SQE may also be implemented across a network, for example, the Internet or may be embedded in another device.

As described with reference to Figure 21, the Data Set Preprocessor 2102 performs two overall functions – building one or more tagged files from the received data set files and dissecting the data set into individual objects, for example, sentences. These functions are described in detail below with respect to Figures 24-26. Although Figures

24-26 present a particular ordering of steps and are oriented to a data set of objects comprising documents and queries comprising sentences, one skilled in the art will recognize that these flow diagrams, as well as all others described herein, are examples of one embodiment. Other sequences, orderings and groupings of steps, and other steps that achieve similar functions, are equivalent to and contemplated by the methods and systems of the present invention. These include steps and ordering modifications oriented toward non-textual objects in a data set, such as audio or video objects.

Figure 24 is an example flow diagram of the steps performed by a build_file routine within the Data Set Preprocessor component of a Syntactic Query Engine. The build_file routine generates text for any non-textual entities within the dataset, identifies document structures (*e.g.*, chapters or sections in a book), and generates one or more tagged files for the data set. In one embodiment, the build_file routine generates one tagged file containing the entire data set. In alternate embodiments, multiple files may be generated, for example, one file for each object (*e.g.*, document) in the data set. In step 2401, the build_file routine creates a text file. In step 2402, the build_file routine determines the structure of the individual elements that make up the data set. This structure can be previously determined, for ,example by a system administrator and indicated within the data set using, for example, HTML tags. For example, if the data set is a book, the defined structure may identify each section or chapter of the book. In step 2403, the build_file routine tags the beginning and end of each document (or section, as defined by the structure of the data set). In step 2404, the routine performs OCR processing on any images so that it can create searchable text (lexical units) associated with each image. In step 2405, the build_file routine creates one or more sentences for each chart, map, figure, table, or other non-textual entity. For example, for a map of China, the routine may insert a sentence of the form,

This is a map of China.

In step 2406, the build_file routine generates an object identifier (*e.g.*, (a Document ID) and inserts a tag with the generated identifier. In step 2407, the build_file routine writes the processed document to the created text file. Steps 2402 through 2407 are repeated for

each file that is submitted as part of the data set. When there are no more files to process, the build_file routine returns.

Figure 25 illustrates an example format of a tagged file built by the build_file routine of the Data Set Preprocessor component of a Syntactic Query Engine. The beginning and end of each document in the file is marked, respectively, with a <DOC> tag 2501 and a </DOC> tag 2502. The build_file routine generates a Document ID for each document in the file. The Document ID is marked by and between a <DOCNO> tag 2503 and a </DOCNO> tag 2504. Table section 2505 shows example sentences created by the build_file routine to represent lexical units for a table embedded within the document. The first sentence for Table 2505,

This table shows the Defense forces, 1996,

is generated from the title of the actual table in the document. The remaining sentences shown in Table 2505, are generated from the rows in the actual table in the document. Appendix B, which is incorporated by reference in its entirety, is a portion of a sample file created by the build_file routine. One skilled in the art will recognize that various processes and techniques may be used to identify documents within the data set and to identify entities (*e.g.*, tables) within each document. The use of equivalent and/or alternative processes and markup techniques and formats, including HTML, XML, and SGML and non-tagged techniques are contemplated and may be incorporated in methods and systems of the present invention.

The second function performed by the Data Set Preprocessor component of the SQE is dissecting the data set into individual objects (*e.g.*, sentences) to be processed. Figure 26 is an example flow diagram of the steps performed by the dissect_file routine of the Data Set Preprocessor component of a Syntactic Query Engine. In step 2601, the routine extracts a sentence from the tagged text file containing the data set. In step 2602, the dissect_file routine preprocesses the extracted sentence, preparing the sentence for parsing. The preprocessing step may comprise any functions necessary to prepare a sentence according to the requirements of the natural language parser component of the ENLP. These functions may include, for example, spell checking, removing excessive

white space, removing extraneous punctuation, and/or converting terms to lowercase, uppercase, or proper case. One skilled in the art will recognize that any preprocessing performed to put a sentence into a form that is acceptable to the natural language parser can be used with techniques of the present invention. In step 2603, the routine sends the preprocessed sentence to the ENLP. In step 2604, the routine receives as output from the ENLP an enhanced data representation of the sentence. In step 2605, the dissect_file routine forwards the original sentence and the enhanced data representation to the Data Set Indexer for further processing. Steps 2601-2605 are repeated for each sentence in the file. When no more sentences remain, the dissect_file routine returns.

The Data Set Indexer (*e.g.*, component 2105 in Figure 21) prepares the enhanced data representations generated from the data set (*e.g.*, the enhanced sentence representation illustrated in Figure 20) to be stored in the data set repository. In one example embodiment, the Data Set Indexer initially stores the enhanced data representation data in generated text files before loading the enhanced data representations into the data set repository, for example, using a bulk loading function of the data set repository. One skilled in the art will recognize that any of a wide variety of well-known techniques may be implemented to load a data set (including the generated enhanced data representations) into the data set repository. For example, another technique for loading writes each record to the data set repository as it is generated instead of writing the records to text files to be bulk loaded.

As described, the SQE uses the ENLP to parse data that is being stored and indexed, as well as to parse queries (*e.g.*, natural language queries) that are submitted against a stored indexed data set. Similar to the preprocessing performed before parsing a data set, the SQE performs preprocessing on submitted queries.

Figure 27 is an example flow diagram of the steps performed by a Syntactic Query Engine to process a natural language query. In step 2701, the Query Preprocessor prepares the natural language query for the ENLP. In step 2702, the ENLP parses the preprocessed natural language query and generates an enhanced data representation of the

query. In step 2703, the Query Builder, generates and executes data queries (*e.g.*, SQL statements) based on the ENLP output (the enhanced data representation).

Figure 28 is an example flow diagram of the steps performed by a preprocess_natural_language_query routine of the Query Preprocessor component of a Syntactic Query Engine. This routine preprocesses the query according to the requirements of the ENLP. Although described with respect to certain modifications to the query, one skilled in the art will recognize that many other preprocessing steps are possible, yield equivalent results, and are contemplated by the methods and systems of the present invention. In step 2801, the routine separates the query into multiple sentences if necessary, based on punctuation (*e.g.*, ";", ":", "?", "!", and ".", where the "." is not part of an abbreviation). In step 2802, the routine removes spaces between any terms that are separated by hyphens. For example, "seventeen – year – old" is converted to "seventeen-year-old". In step 2803, the routine removes extraneous spaces from the query, leaving the words separated by one space. In step 2804, the routine spell-checks the routine. In one embodiment, the routine automatically corrects any detected spelling errors. In another embodiment, the routine requests an indication of whether and how each spelling error is to be corrected.

The enhanced data representations describe one or more ways in which the meaningful terms of an object (*e.g.*, a sentence) may be related. (See description with reference to Figure 20.) As described earlier, enhanced data representations of the data in the data set are stored in the data set repository, and enhanced data representations are used to generate data queries when the SQE processes a natural language query. The enhanced data representations describe the relationships between meaningful terms within each sentence, as determined by the ENLP. Each meaningful term may be associated with one or more syntactic or grammatical roles. Redundancy is not discouraged, because it may yield additional results. For example, a term may be part of two described relationships that are associated with different grammatical roles, *e.g.*, a term may appear as both an "object" and a "noun modifier." The relationships that are selected and represented are heuristically determined as those relationships that will tend to generate additional relevant

results. A current embodiment uses the specific relationships described in Figure 20 and shown as stored in the data repository in Figure 29 and described by Figures 31-36. However, one skilled in the art will recognize that other relationships and grammatical roles may be described in the enhanced data representation, and the determination of these roles and relationships relates to the types of objects in the data set.

Figure 29 is an example block diagram showing the structure of an example Data Set Repository of a Syntactic Query Engine. The set of stored tables represent the roles and relationships between the determined meaningful terms for each parsed sentence of each document in the data set, as determined by the ENLP, and correspond to the enhanced data representations generated. The Subject Table 2901 stores one record for each determined subject/verb combination in each parsed sentence in each document. The Object Table 2902 stores each determined object/verb combination for each parsed sentence in each document. The Subject_Object table stores each identified subject/verb/object combination for each parsed sentence. In an alternate embodiment, the Subject_Object table is implemented as a view that is a dynamically maintained join between the subject and object tables, joined on the verb, Document ID, and Sentence ID fields. The Preposition table 2903 stores each verb/preposition/verb modifier combination for each parsed sentence in each document. The Noun_Modifier table 2904 stores each noun/noun modifier combination in each parsed sentence in each document. A noun modifier may be a noun, a noun phrase, or an adjective. The Sentence table 2905 stores the actual text for each sentence in each document. In addition, the Sentence table 2905 stores the Governing Verb, Related Subject, and Related Object of each sentence, as identified by the postprocessor component of the ENLP. The governing verb is the main verb in a sentence. The related subject and related object are the subject and object, respectively, related to the governing verb. For example, in the sentence

The girl walks the dog.

"walks" is the governing verb, "girl" is the related subject, and "dog" is the related object. These fields may be left blank if the postprocessor is unable to determine the governing verb, related subject, or related object. The Date, Money Amount, Number, Location,

Person, Corporate Name, and Organization fields of the Sentence table 2905 store binary indicators of whether or not the sentence contains a term that the SQE recognizes as an attribute of the attribute type indicated by the field name. The Attributes table 2906 is an optional table that stores the values of specific data types found within each document. As described above, attributes are settable parameters and may include, for example, names of countries, states, or regions, document sections, and dates. As described with respect to Figure 15, these attributes may be used by the SQE to filter data query results. The optional Parent table 2907 is used to indicate a hierarchical structure of objects in the data set. This allows a section, subsection, chapter, or other document portion to be identified by the Data Set Indexer component of a Syntactic Query Engine as a "document", while storing the relationships between multiple documents or document portions in a hierarchical fashion.

Figure 30 is an example flow diagram of the steps performed by a parse_sentence routine of the Enhanced Natural Language Parser component of a Syntactic Query Engine. In summary, the routine parses the designated sentence or phrase, identifies the grammatical roles of terms within the sentence, generates an enhanced data representation of the sentence, and constructs an output string. In step 3001, the natural language parser component of the ENLP parses the preprocessed sentence. In step 3002, the parse_sentence routine calls the determine_grammatical_roles subroutine (discussed in detail with reference to Figure 31) to determine the grammatical roles of and relationships between terms within the sentence. In step 3004, the routine calls the construct_output_string routine (discussed in detail with reference to Figure 37) to format the generated enhanced data representation for further processing.

Figure 31 is an example flow diagram of the steps performed by a determine_grammatical_roles subroutine within the parse_sentence routine of the Enhanced Natural Language Parser. In summary, the routine converts terms, as appropriate, to a standard form (*e.g.*, converting all verbs to active voice), identifies attributes (*e.g.*, names of countries) within the sentence, determines the grammatical roles of meaningful terms in the sentence, and initiates the generation of an enhanced data

representation. In step 3101, the routine converts each word that is identified as a subordinate term to an associated governing term. Subordinate terms and governing terms are terms that are related in some way, similar to multiple tenses of a verb. For example, the governing term "Ireland" is associated with subordinate terms "Irish," "irish," "Irishman," "irishman," "Irishwoman," "irishwoman," and "ireland." Converting subordinate terms to an associated governing term ensures that a standard set of terms is used to represent data, for example relating multiple terms to a country, as shown in the example above, thus increasing potential contextual matches. Subordinate terms that are identified as occurring within a noun phrase, for example, "Korean," occurring with the noun phrase "Korean War," are not converted to the associated governing term to preserve the specific meaning of the noun phrase. For example, "Korean War" has a specific meaning that is not conveyed by the terms "Korea" and "war" independently. Appendix C, which is herein incorporated by reference in its entirety, is an example list of subordinate terms and their associated governing terms for an example SQE. Typically, this information is stored by the SQE as a configurable list that is related to the data set, preferably initialized when the SQE is installed, for example by a system administrator.

In step 3102, the determine_grammatical_roles routine converts the verbs within the sentence to active voice. This ensures that the SQE will identify data within the data set in response to queries regardless of verb tense. (See the example described with reference to Figure 1.) In step 3103, the routine identifies attribute values, which can later be used to filter search results.

Steps 3104-3111 are described with reference to an example query,

Does Argentina import or export natural gas from the south of Patagonia? (Q)

Figure 36A is a graphical representation of an example parse tree generated by a natural language parser component of an Enhanced Natural Language Parser. The example parse corresponds to this query. An enhanced data representation of the example query is shown in Figure 36B.

In steps 3104-3111, the determine_grammatical_roles routine builds the data structures that correspond to the enhanced data representation. Specifically, in step 3104, the routine generates Subject data structures identifying terms that may be the subject of the sentence (similar to the Subject table described with reference to Figure 29). For example, given query Q above, Table 1 shows the generated Subject structures.

Table 1

Subject	Verb
Argentina	import
Argentina	export

The steps performed in generating the Subject structures are described in detail with reference to Figure 32. In step 3105, the routine generates Object data structures identifying terms that may be the object of the sentence (similar to the Object table described with reference to Figure 29). For example, given query Q above, Table 2 shows the generated Object structures.

Table 2

Verb	Object
import	gas
import	natural gas
export	gas
export	natural gas

The steps performed in generating the Object structures are described in detail with reference to Figure 33. In step 3106, the routine generates Preposition structures that are similar to the structure of the Preposition table described with reference to Figure 29. For example, given query Q above, Table 3 shows the generated Preposition structures.

Table 3

Verb	Preposition	Modifier
import	from	south
import	from	Patagonia
export	from	south
export	from	Patagonia

In step 3107, the routine generates Subject/Object structures to represent the ways in which terms that are identified as potential subjects and objects of a sentence may be related. Each subject is paired with each object that is associated with the same verb. In addition, each subject is paired with each modifier in the Preposition structure that is associated with the same verb. For example, given query Q above, Table 4 shows the generated Subject/Object structures.

Table 4

Subject	Object
Argentina	gas
Argentina	natural gas
Argentina	south
Argentina	Patagonia

In step 3108, the routine generates Subject/Modifier structures to describe the relationships between related nouns, noun phrases, and adjectives. The generate_subject_modifier routine is described in detail with reference to Figure 34.

In step 3109, the routine determines whether or not the sentence being parsed is a query (as opposed to an object of a data set being parsed for storing and indexing). If the designated sentence or phrase is a query, the routine continues in step 3110, else it returns. In steps 3110 and 3111, the routine generates Generalized Subject/Object structures and Generalzied Subject/Modifier structures. These structures may be used by the Query Builder to generate additional data queries to return results that may be contextually relevant to the submitted natural language query. The

generate_generalized_subject_object routine is described in detail with reference to Figure 35. The Generalized Subject/Modifier structures are generated from previously generated Subject/Object structures. The object is designated as the subject in the new Generalized Subject/Modifier structure, and the subject is designated as the modifier in the new Generalized Subject/Modifier structure.

Figure 32 is an example flow diagram of the steps performed by a generate_subject_structure subroutine of the determine_grammatical_roles routine. This routine identifies, for each verb in the sentence, each noun, noun phrase, or adjective that may be a subject associated with the designated verb and stores it in a Subject structure. In step 3201, the routine searches for all verbs in the syntactic data representation (*e.g.*, a parse tree) generated by the natural language parser. In step 3202, the routine sets the current node to an identified verb. In steps 3203-3206, the routine loops searching for all terms that are potentially subjects related to the identified verb. Specifically, in step 3203, the routine moves toward the beginning of the sentence (*e.g.*, to the next leaf to the left in the parse tree from the current node). In step 3204, the routine examines the node and determines whether or not it is a noun, a noun phrase, or an adjective. If the routine determines that the current node is a noun, a noun phrase, or an adjective, then it is identified as a subject and the routine continues in step 3205, else it continues in step 3206. In step 3205, the routine creates a Subject structure identifying the current node as the subject and the previously identified verb (the current verb) as the verb. Additionally, if the current node is a noun phrase, the routine creates a Subject structure identifying the tail of the noun phrase (*e.g.*, "gas" of the noun phrase "natural gas") as a subject associated with the current verb. The routine then continues searching for additional subjects related to the current verb by looping back to step 3203. In step 3206, the routine examines the current node and determines whether or not it is the left-most leaf or a verb from a different verb phrase. If the current node is not the left-most leaf or a verb from a different verb phrase, the routine continues searching for additional subjects related to the current verb by returning to the beginning of the loop, in step 3203, else it continues in step 3207. In step 3207, the routine determines whether or not all of the identified verbs have been processed.

If there are more verbs to process, the routine continues in step 3202 and begins another loop with the new verb as the current node, otherwise it returns.

Figure 33 is an example flow diagram of the steps performed by a generate_object_structure subroutine of the determine_grammatical_roles routine. This routine identifies, for each verb in the sentence, each noun, noun phrase, or adjective that may be an object associated with the designated verb and stores it in an Object structure. In step 3301, the routine searches for all verbs in the syntactic data representation generated by the natural language parser. In step 3302, the routine sets the current node to an identified verb. In steps 3303-3306, the routine loops searching for all terms that are potentially objects related to the identified verb. Specifically, in step 3303, the routine moves toward the end of the sentenct (*e.g.*, to the next leaf to the right in the parse tree from the current node). In step 3304, the routine examines the node and determines whether or not it is a noun, a noun phrase, or an adjective. If the routine determines that the current node is a noun, a noun phrase, or an adjective, then it is identified as an object and the routine continues in step 3305, else it continues in step 3306. In step 3305, the routine creates an Object structure identifying the current node as the object and the previously identified verb (the current verb) as the verb. Additionally, if the current node is a noun phrase, the routine creates an Object structure identifying the tail of the noun phrase (*e.g.*, "gas" of the noun phrase "natural gas") as the object associated with the current verb. The routine then continues searching for additional objects related to the current verb by looping back to step 3303. In step 3306, the routine examines the current node and determines whether or not it is the right-most leaf, a verb from a different verb phrase, or a preposition other than "of." If the current node is not the right-most leaf, a verb from a different verb phrase, or a preposition other than "of," the routine continues searching for additional objects related to the current verb by returning to the beginning of the loop in step 3303, else it continues in step 3307. In step 3307, the routine determines whether or not all of the identified verbs have been processed. If there are more verbs to process, then the routine continues in step 3302, and begins another loop with the new verb as the current node, otherwise it returns.

Figure 34 is an example flow diagram of the steps performed by a generate_subject_modifier subroutine of the determine_grammatical_roles routine. This routine identifies, for each noun in the sentence, each other noun or adjective that may be related to the designated noun and stores it in a Subject/Modifier structure. In step 3401, the routine searches for all nouns and noun phrases in the syntactic data representation (*e.g.*, a parse tree) generated by the natural language parser. In step 3402, the routine sets the current node to an identified noun or noun phrase. In steps 3403-3406, the routine loops searching for all terms that may be related to the identified noun. Specifically, in step 3403, the routine moves toward the beginning of the sentenct (*e.g.*, to the next leaf to the left in the parse tree from the current node). In step 3404, the routine examines the node and determines whether or not it is a noun, a noun phrase, or an adjective. If the routine determines that the current node is a noun, a noun phrase, or an adjective, then it is identified as a modifier and the routine continues in step 3405, else it continues in step 3406. In step 3405, the routine creates a Subject/Modifier structure identifying the current node as the modifier and the previously identified noun as the subject. Additionally, if the current node is a noun phrase, the routine creates a Subject/Modifier structure identifying the tail of the noun phrase (*e.g.*, "gas" of the noun phrase "natural gas") as the modifier associated with the current noun. The routine then continues searching for additional modifiers related to the current noun by looping back to step 3403. In step 3406, the routine examines the current node and determines whether or not it is the preposition, "of." If the current node is the preposition "of," the routine continues searching, by returning to the beginning of the loop, in step 3403, else it continues in step 3407. In step 3407, the routine determines whether or not all of the identified nouns have been processed. If there are more nouns to process, the routine continues in step 3402 and begins another loop with the new noun as the current node, otherwise it returns.

As described with reference to Figure 31, the determine_grammatical_roles routine calls the generate_generalized_subject_object and generate_generalized_subject_modifier subroutines (steps 3110 and 3111 of Figure 31)

when the sentence being parsed is a query instead of an object in a data set being indexed for storage.

Figure 35 is an example flow diagram of the steps performed by a generate_generalized_subject_object subroutine of the determine_grammatical_roles routine. This routine identifies, for each noun in the sentence, each other noun or adjective that may be an object related to the designated noun and stores it in a Subject/Object structure. In step 3501, the routine searches for all nouns and noun phrases in the syntactic data representation (*e.g.*, a parse tree) generated by the natural language parser. In step 3502, the routine sets the current node to an identified noun (or noun phrase). In steps 3503-3509, the routine loops searching for all terms that may be an object related to the identified noun. Specifically, in step 3503, the routine moves toward the end of the sentence (*e.g.*, to the next leaf to the right in the parse tree from the current node). In step 3504, the routine examines the node and determines whether or not it is a preposition other than "of." If the routine determines that the current node is a preposition other than "of," then it continues in step 3505, else it continues in step 3508. In step 3508, the routine examines the node and determines whether or not it is a verb or the last node. If the routine determines that the current node is a verb or the last node, then it continues in step 3510, else it continues searching, by returning to the beginning of the loop, in step 3503. In step 3505, the routine moves toward the end of the sentence (*e.g.*, to the next leaf to the right in the parse tree from the current node). In step 3506, the routine examines the current node and determines whether or not it is a noun, a noun phrase, or an adjective. If the routine determines that the current node is a noun, a noun phrase, or an adjective, then it is identified as an object and the routine continues in step 3507, else it continues in step 3509. In step 3507, the routine creates a Generalized Subject/Object structure identifying the current node as the object and the previously identified noun as the subject. Additionally, if the current node is a noun phrase, the routine creates a Subject/Object structure identifying the tail of the noun phrase (*e.g.*, "gas" of the noun phrase "natural gas") as the object associated with the current "subject" node. After creating one or more Generalized Subject/Object structures, the routine continues looping in step 3505, looking for any other

nouns, noun phrases, or adjectives to relate to the identified "subject" noun. In step 3509, the routine determines whether or not the current node is the preposition "of." If the current node is the preposition "of," then the routine continues looping in step 3505, else it continues in step 3510. In step 3510, the routine determines whether or not all of the identified nouns and noun phrases have been processed. If there are more nouns or noun phrases to process, the routine continues in step 3502 and begins another loop with the new noun as the current node, otherwise it returns.

Figure 36B is an illustration of an enhanced data representation of an example natural language query generated by an Enhanced Natural Language Parser. The natural language query,

> Does Argentina import or export natural gas from the south of Patagonia?

(query Q) is described by the roles and relationships shown in rows 1-28. Rows 1 and 2 are generated by the generate_subject_structure subroutine described with reference to Figure 32. Rows 3-6 are generated by the generate_object_structure subroutine described with reference to Figure 33. Rows 7-10 are generated by the generate Preposition structures routine described with reference to step 3106 of Figure 31. Rows 11-14 are generated by the generate Subject/Object structures routine described with reference to step 3107 of Figure 31. Row 15 is generated by the generate Subject/Modifier structures routine described with reference to step 3108 of Figure 31. Rows 16-20 are generated by the generate_generalized_subject_object routine described with reference to Figure 35. Specifically, rows 16-19 are generated in step 3507 of Figure 35. Row 20 is generated in step 3511 of Figure 35. Rows 21-28 are generated by the generate Generalized Subject/Modifier structures routine described with reference to step 3111 of Figure 31.

At this point, the ENLP has determined the syntax, grammatical roles, and relationships between terms of the sentence. The ENLP has also generated an enhanced data representation for the sentence with all of the structures described with reference to Figure 31. The ENLP next constructs an output string (step 3003 of Figure 30) that will be used by the Data Indexer to index and store the enhanced data representation when the

SQE is processing an object of the data set, or by the Query Builder to generate data queries that will be executed against the data set repository.

An output string generated by an example ENLP is of the form:

{Parameter String};{Parameter String};...;{Parameter String}|{Attribute List};{Verb List};{Word List};{ Sentence}

Each {Parameter String} is a set of six single quoted parameters, separated by semi-colons, of the form:

'subject';'verb';'preposition';'verb modifier';'object';'noun modifier'

where a wildcard character, for example "#" or "*" may be substituted for one or more of the parameters.

The {Attribute List} is a list of <Attribute Name;Attribute Value> pairs, separated by semi-colons, for example:

country;France;country;Japan

The {Verb List} is a list of all of the identified verbs in the sentence separated by semi-colons. It includes even the verbs that appear in the {Parameter String} parameters.

The {Word List} is a distinct list of all the words found in the Parameter Strings that are not also Attribute Values in the Attribute List separated by semi-colons.

The {Sentence} is the sentence that the ENLP processes after any preprocessing that may include modifying the text case and correcting spelling.

Figure 37 is an example flow diagram of the steps performed by a construct_output_string routine of the Enhanced Natural Language Parser. This routine sorts the generated parameter sets that comprise the enhanced data representation of the natural language query based on which sets are most likely to generate data queries with contextually accurate results. The routine then constructs an output string of the format described. In step 3701, the routine generates parameter strings from the structures generated as described with reference to steps 3104-3111 of Figure 31. In step 3702, the routine sorts the generated parameter strings, preferably in order of increasing ambiguity of terms, such that the parameter strings comprising terms with less ambiguity (more apt to return contextually accurate results) rank higher than those comprising terms with more

ambiguity (less apt to return contextually accurate results). Although any method or combination of methods may be used to order/sort the generated parameter strings, two example sorting methods assign a weight to each generated parameter string based on a determination of term ambiguity.

Using the first method, the SQE assigns a weight to each parameter string based on a, preferably previously stored, Inverse Document Frequency ("IDF") of each parameter. The IDF of a particular parameter is equal to the inverse of the number of times that particular term appears in the data set. For example, a word that appears only one time in a data set has an IDF value of 1 (1 divided by 1), while a word that appears 100 times in a data set has an IDF value of 0.01 (1 divided by 100). In one embodiment, the weight assigned to a parameter string is equal to the sum of the IDF values of each parameter in the string. Terms that are not found in the data set are assigned an IDF of –1. In this embodiment, the parameter strings comprising the terms that appear least frequently in the data set are given a higher weight because they are most apt to return results pertinent to the natural language query.

A second method may be employed by an example SQE to weight the parameter strings according to the polysemy of each parameter. The polysemy of a term is the number of meanings that the term has. The weight assigned to each parameter string is equal to the inverse of the sum of the polysemy values for each parameter in the string. Polysemy values may be obtained for example from a dictionary service, an example of which is WordNet. One skilled in the art will recognize that any such mechanism for determining polysemy values may be used. According to convention, the minimum polysemy value is 1 (indicating that a word has only one meaning). In one embodiment of the SQE, if a polysemy value cannot be determined for a word, the SQE assigns it a polysemy value of 0.5 (indicating that the word likely has a context-specific meaning). In this embodiment, the parameter strings comprising the terms with the least ambiguity of meaning (the lower polysemy values) are given a higher weight because they are most apt to return pertinent results to the natural language query.

In step 3703, the routine adds the ordered parameter strings to the output string. When used to parse a query (not to index an object of the data set) these parameter strings are subsequently used by the Query Builder to generate data queries (*e.g.*, SQL queries). In an alternate embodiment, in order to limit the number of data queries that will be generated, a subset of the ordered parameter strings are added to the output string. In one embodiment, the first *n* parameter strings are added where *n* is a configurable number. In another embodiment, when a weight is assigned to each parameter string, a percent value of the maximum assigned weight may be used to limit the parameter strings that are included in the output string. For example, if the highest weight assigned is 10, the SQE may use 70% as a limit, including in the output string only those parameter strings that are assigned a weight of at least 7 (*i.e.*, 70% of the maximum assigned weight). One skilled in the art will recognize that any limiting technique may be used to restrict the number of parameter strings included in the output string if it is desirable to limit the number of data queries. In step 3704, the routine adds any identified attribute values to the output string, according to the described output string format. The identified attribute values may be used by the Query Builder to filter the data query results. In step 3705, the routine adds a list of identified verbs to the output string, according to the described output string format. In step 3706, the routine adds to the output string a list of the words that are present in the parameter strings and are not in the list of attribute values. In step 3707, the routine adds the sentence to the output string. Steps 3705-3707 are performed when processing a query to add additional data to the output string that may be used by the Query Builder to generate additional data queries in the event that the data queries generated based on the standard parameter sets do not yield a sufficient number of results.

When the SQE is indexing a data set, the described ENLP output is forwarded to the Data Indexer Component of the SQE to be stored in the data set repository. Figure 38 is an example flow diagram of the steps performed by an index_data routine of the Data Indexer component of a Syntactic Query Engine. In step 3801, the routine creates one text file for each table in the data set repository. In step 3802, the routine assigns a Sentence identifier (*e.g.* Sentence ID) to the parsed sentence. In step

3803, the routine writes data, as appropriate, to each text file based on the received ENLP output. The described output parameter strings are used to populate the text files that are bulk loaded into the data repository. Steps 3802 and 3803 are repeated for each enhanced data representation received from the ENLP. The specific format of the text files is typically dictated by the bulk loading requirements of the software used to implement the data set repository.

When the SQE is processing a query (as opposed to indexing a data set), after parsing the query and forwarding the ENLP out to the Query Builder, the Query Builder generates and executes data queries against the data repository. Figures 39A and 39B are example flow diagrams of the steps performed by a build_query routine within the Query Builder component of a Syntactic Query Engine. The routine executes once for each designated parameter string (output by the ENLP) and generates and executes one or more data queries against the data set repository based on the values of the parameters that make up the designated parameter string.

Specifically, in step 3901, the build_query routine examines the designated parameters to make a preliminary determination regarding the source of the natural language query. As described with reference to Figures 4 and 10, respectively, the SQE supports both general searches and advanced searches. In the example embodiment, general searches return sentences; advanced searches return a list of verbs or other designated parts of speech or grammatical roles that can be further used to search for sentences. The steps performed by the build_query routine differ depending on the source of the natural language query. The advanced search functionality of the example embodiment described allows natural language queries to be submitted that designate a subject and/or an object and retrieve one or more verbs. The steps performed by the build_query routine in alternate embodiments may vary depending on the syntactic and grammatical roles that can be designated within the advanced search functionality. One skilled in the art will understand how to modify the routine for a specific SQE implementation. If the designated parameter string has parameters that are only a subject and/or an object (with all of the other parameters wildcards), the routine continues in step

3904, else it continues in step 3902 to build and execute a data query, having determined that the natural language query originated from the general search functionality. In step 3902, the build query routine builds and executes a single data query based on the parameters in the designated parameter string and returns resultant sentences from the data set repository. In step 3904, if the parameters of the designated string are only a subject and/or an object, then the routine builds and executes a data query, based on the subject and/or object parameters, that returns a related verb list. The related verb list comprises distinct verbs and includes a frequency count of the number of times each verb appears in sentences within the data set. The routine temporarily stores the related verb list for later use. In step 3905, the routine determines whether or not the natural language query was submitted as an advanced search, and, if so, continues in step 3906, else continues in step 3909. In step 3906, the routine determines whether or not both a subject and an object are designated. If both are designated, the routine continues in step 3907, else it returns the related verb list (the results of the query executed in step 3904). In step 3907, the routine builds and executes a data query, based on the designated subject and object in *reverse* roles (*i.e.*, the designated subject as the object and the designated object as the subject), that returns a related verb list (as described in step 3904), and returns the related verb lists resulting from steps 3904 and 3907.

If, in step 3905 the routine determines that the natural language query was not submitted through an advanced search, then in steps 3909-3917 the routine generates and executes a series of data queries that attempt to locate objects in the data set in a heuristic manner based upon the subject and/or object of the designated parameter string and verbs that are similar to the verbs specified in the natural language query. In particular, the routine generates and executes data queries based upon the designated subject and/or object in combination with (1) the requested verbs; (2) verbs that are entailed from the requested verbs; and (3) verbs that are related to the requested verbs, such as those produced in the related verb list resulting from step 3904. If these queries do not generate sufficient results (which is preferably modifiable), then the routine executes the same set of data queries with the subject and/or object appearing in *inverse* grammatical

roles. In addition, weights are associated with the resultant objects (*e.g.*, sentences) to indicate from which data query they came, so that the overall result output can be ordered in terms of what results are most likely to address the initial query. These weights are preferably configurable, for example, by a system administrator of the SQE.

Specifically, in step 3909, the routine builds and executes a data query using the designated subject and/or object and any verb that appears in the initial (natural language) query (a requested verb). Because a search was already performed for verbs that correspond to the designated subject and/or object in step 3904, the results of the step 3904 data query can be used to streamline the data query of step 3909. In particular, a query is preferably generated and executed using the verbs that appear in both the {Verb List} of the output string generated by the ENLP and the related verb list returned by the query in step 3904 (the intersection of these two lists). These comprise the verbs that are in the initial query that have also been found to be present in the data set in objects that contain a similar subject and/or object. (Since the verbs in the {Verb List} include all of the verbs present in the natural language query, any designated parameter string that also includes a verb as one of the parameters will be accounted for also in the {Verb List}.) The routine associates a default weight with the resulting sentences indicating that these sentences came from a match of verbs present in the initial query.

In step 3910, the routine builds and executes a data query using the designated subject and/or object and verbs that are entailed from the requested verbs (verbs that appear in the initial query). As in step 3909, the results of the data query of step 3904 can be used to streamline this data query. In particular, a query is preferably generated and executed using the verbs that entail each of the verbs that appear in both the {Verb List} of the output string generated by the ENLP and the related verb list returned by the query in step 3904 (the intersection of these two lists). Entailed verbs are available through existing applications, for example, WordNet, and are verbs that, based on meaning, are necessarily required prior to an action described by another verb. For example, given the verb, "snore," "sleep" is an entailed verb because (typically) sleeping occurs prior to snoring. The routine associates a weight referred to as an "entailed weight" with the

resulting sentences indicating that these sentences came from a match of verbs that entail from verbs present in the initial query.

In step 3911, the routine builds and executes a data query using the designated subject and/or object and verbs that are related to the requested verbs (verbs that appears in the initial query). In one embodiment, the related verbs are verbs that appear in the related verb list returned by the query in step 3904 that are *not* in the {Verb List} of the output string generated by the ENLP. These are the verbs that are present in the data set in objects that contain a similar subject and/or object and that are not also requested verbs. The routine associates a weight referred to as a "verb similarity weight" with each of the resulting sentences indicating that these sentences came from a match of verb that is related to a verb present in the initial query. In one embodiment, the verb similarity weight differs with each related verb and is a configurable weighting of relative weights assigned by some external application. For example, a dictionary such as WordNet can be used to associated a "similarity measure" with all of the verbs present in a data set. These similarity measures can be further weighted by a multiplier to generate weights that indicate that the resulting sentences are less useful than those returned from data queries involving requested verbs or entailed verbs. In an alternate embodiment, the different weights for resulting sentences are determined by another application, for example, WordNet.

In step 3912, the routine determines whether or not the number of results returned by the data queries generated in steps 3909-3911 is greater than k, where k is preferably a configurable number. If the number of results is greater than k, the routine returns the results determined thus far, else it continues in step 3914. In steps 3914-3917, the routine generates and executes data queries that are the same as those of corresponding steps 3909-3911, with the exception that the roles of the designated subject and designated object are reversed. That is, the designated subject becomes the object and the designated object becomes the subject in the generated data query. Weights are also assigned accordingly to the resulting sentences. As discussed with reference to Figure 14, querying the data set using the inverse subject/object relationship may return additional, contextually

accurate, results that may not be returned when using the original subject/object relationship. After executing these queries, the resulting weighted sentences are returned.

In some embodiments, the results of the natural language query are sorted when they are returned. One skilled in the art will recognize that any sorting method may be used to sort the query results. In one embodiment, the results are first sorted based on the weights (default, entailed, and verb similarity weights) returned with the results as described with reference to Figures 39A and 39B. Next, the results are sorted based on attribute values designated, for example, by a user. For example, if a user specifies the name of a country in a natural language query, resulting sentences that also contain the specified country name are ranked higher than resulting sentences that do not. Finally, the resulting sentences that are returned by data queries generated from more than one parameter string are ranked higher than those from a single data query. Other arrangements and combinations are contemplated.

Although specific embodiments of, and examples for, methods and systems of the present invention are described herein for illustrative purposes, it is not intended that the invention be limited to these embodiments. Equivalent methods, structures, processes, steps, and other modifications within the spirit of the invention fall within the scope of the invention. The various embodiments described above can be combined to provide further embodiments. Also, all of the above U.S. patents, patent applications and publications referred to in this specification, including U.S. Provisional Application No. 60/312,385, filed on August 14, 2001, and entitled "Method and System for Enhanced Data Searching" are incorporated herein by reference, in their entirety. Aspects of the invention can be modified, if necessary, to employ methods, systems and concepts of these various patents, applications and publications to provide yet further embodiments of the invention. In addition, those skilled in the art will understand how to make changes and modifications to the methods and systems described to meet their specific requirements or conditions. For example, the methods and systems described herein can be applied to any type of search tool or indexing of a data set, and not just the SQE described. In addition, the techniques

described may be applied to other types of methods and systems where large data sets must be efficiently reviewed. For example, these techniques may be applied to Internet search tools implemented on a PDA, web-enabled cellular phones, or embedded in other devices. Furthermore, the data sets may comprise data in any language or in any combination of languages. In addition, the user interface components described may be implemented to effectively support wireless and handheld devices, for example, PDAs, and other similar devices, with limited screen real estate. These and other changes may be made to the invention in light of the above-detailed description. Accordingly, the invention is not limited by the disclosure.